

Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

### **REMARKS**

In this amendment no claims are cancelled or added . Accordingly, claims 1-18 and 34-59 are pending.

#### **1. *Williams Declaration***

As a preliminary matter, Applicants wish to address the declaration filed under 37 C.F.R. §1.132 by Steve Williams filed on September 7, 2004. The Office Action states that the Williams Declaration is insufficient to overcome the rejection of claims 1, 34 and 54 for two reasons. First, the Office Action asserts that the Williams Declaration is not signed by all inventors. Second, the Office Action asserts that the Declaration "lacks technical validity". These assertion do not justify the Office Action's summary dismissal of the Williams Declaration.

Regarding the first point, Applicants are aware of no requirement, in the MPEP or in the CFR, that a **§ 1.132 Affidavit** must be signed by all inventors in order to be considered. The Examiner is invited to cite appropriate authority for this assertion or withdraw this objection.

Regarding the second point, its rationale is ***expressly prohibited*** in § 716.01(B) of the MPEP. Accordingly, this objection must be withdrawn.

Applicants assert that the Williams Declaration provides persuasive evidence that all of the pending claims are patentably distinct from Koshisaka. Furthermore, there is no justifiable reason to disregard the Williams Declaration. Proper consideration of the Declaration in conjunction with the arguments presented herein is earnestly solicited.

#### **2. *Rejection of claims 1-3, 9-12, 15 and 54-57 under 35 U.S.C. §103 over Koshisaka et al.***

Claims 1-3, 9-12, 15 and 54-57 are rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,629,109, issued to Koshisaka *et al.* (hereinafter Koshisaka). See Office Action, at Page 3. This rejection is respectfully traversed.

Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

In order to establish a *prima facie* case of obviousness, the Office Action must 1) determine the scope and content of the prior art, 2) determine the level of ordinary skill in the art, 3) determine the differences between the prior art and the claimed invention and 4) evaluate evidence of secondary considerations. Here the Official Action has erred in carrying at least points 1 and 3.

#### **Koshisaka Discloses API That is not an Operating System**

The Office Action has fundamentally misconstrued the scope and meaning of the term API as used in Koshisaka. Nothing in Koshisaka would have suggested to a person of ordinary skill in the art that the Koshisaka's API is an operating system as asserted in the Office Action. Koshisaka clearly distinguishes between operating systems and application programs. See elements 3 and 1 of Figure and Column 7, lines 3-12, for example. They are separate and distinct aspects of the Koshisaka device.

In addition, one of the other references of record, Dunphy, U.S. Patent No. 5,638,509 (hereinafter Dunphy) discloses an operating system 19 and separate and distinct application programs 8. See Column 3, lines 36-47 and Figure 1. Thus, it is readily apparent that the prior art does not equate application programs and operating systems as alleged in the Office Action.

Moreover, applicable technical dictionaries recognize the difference between APIs and operating systems. The Webster's New World Dictionary of Computer Terms defines "API" as a set of standards or conventions by which programs can call specific operating system or network services. The same dictionary defines "Operating System" as a master control program that manages the computer's internal functions, such as accepting keyboard input, and that provides a means to control the computer's operations and file system. See pages 33 and 338 of the Dictionary attached hereto as Exhibit 1.

In sum, the evidence of record shows how a person of ordinary skill in the art would have construed the term "operating system". The evidence of record overwhelmingly urges the conclusion that the API of Koshisaka is materially different from an operating system.

Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

**The Differences Between Koshisaka and the Instant Claims are Significant**

There are several significant differences between Koshisaka and the subject matter of claims 1-3, 9-12, 15 and 54-57. One difference is that those claims recite "detecting an instruction by an operating system to perform an operation on an operating file". Notwithstanding the comments to the contrary in the Office Action, Koshisaka does not teach or disclose this detecting step.

As mentioned above, Koshisaka errs in equating its API with the operating system of the claims. The term "operating system" is defined in the instant specification at paragraph 28. Where an explicit definition is provided by the applicant for a term, that definition will control interpretation of the term as it is used in the claim. *Toro Co. v. White Consolidated Industries Inc.*, 199 F.3d 1295, 1301, 53 USPQ2d 1065, 1069 (Fed. Cir. 1999). Here, this axiom is particularly applicable as the definition of the term "operating system" set forth in the specification parallels the understanding of the skilled artisan.

As recognized by the Office Action at page 2, given the proper interpretation of the term "operating system", the subject matter of claim 1 would not have been obvious over Koshisaka to a person of skill in the art.

Koshisaka, teaches "an application-centric" file revision management system and method by which file revision management allegedly can be implemented even if applications are not provided with file revision management functions. Koshisaka teaches the use of an Applications Programming Interface (API) layer to be placed between an application and an operating system. According to Koshisaka, file backup reliability of the applications can be improved by this file revision management system. See Koshisaka, column 1, lines 57-63. See also Williams declaration, Paragraph 5. A file manipulation monitoring section in Koshisaka first detects file manipulation (e.g., file deletion or file name change) that is to be executed by the application by intercepting these instructions as they are generated by the application. See Williams declaration, Paragraph 6. Koshisaka specifically states, "the file manipulation monitoring section constantly monitors API (Application Program Interface) commands which are outputted by the application 1 to the operating system and thereby detects the file manipulation

Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

which is (going to be) executed by the application 1." See Koshisaka, column 6, lines 34-38.

The method of the present invention, as defined by claims 1 and 54, is a "data-centric" approach to file archiving, not an application-centric approach. This distinction is evidenced by the claim limitations of, "detecting an instruction *by an operating system* to perform an operation on an operating file," and "capturing the operating file temporally proximate to the operation being performed on the operating file, responsive to the detection of the instruction."

In direct contrast, Koshisaka is not data-centric; that is, it does not teach detection of an instruction by an operating system, which is independent of application. Rather, the detected *instruction* or command in Koshisaka is *based on the API command* and is *by the application*, not *by the operating system*. See Williams declaration, Paragraphs 5, 6 and 7. For example, in Koshisaka, when an API command requesting file deletion is outputted by the application, the command is detected and hooked by the file manipulation monitoring section in the API. Subsequently, the processing section sends a different API command to the operating system. In other words, the instruction is first detected by the API and hooked. After the instruction is hooked, a different instruction is passed to the operating system, and the original command sent by the API to the operating system is not executed during performance of Koshisaka's revision management system activities. See Williams declaration, Paragraph 5. In contrast, the present invention does not pass a different set of commands to the operating system, but rather performs its independent function prior to allowing the **originally requested** operating system commands to process.

As a result of detecting the instruction *by the application*, unlike the present invention, it is believed that Koshisaka provides a very limited layer of file protection, as file protection occurs only if the application is compatible with the Koshisaka assumptions of API behavior (Koshisaka, column 7, lines 59-64). See Williams declaration, Paragraph 8. Furthermore, the Examiner concedes that Koshisaka does not teach capturing the operating file and moving this captured file to an alternate storage location responsive to the detection of the instruction. The present invention differs completely in both design and implementation from Koshisaka.

Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

In view of the foregoing, the subject matter of claims 1 and 54 would not have been obvious in view of Koshisaka. It follows that claims 1 and 54 are allowable. Claims 2, 3, 9 and 15 depend from independent claim 1. Thus, by definition, claims 2, 3, 9 and 15 are allowable over Koshisaka for at least the reasons offered with respect to claim 1. Claims 55-57 depend from claim 54. Thus, these claims are also allowable over Koshisaka for at least the reasons offered with respect to claim 54.

***Rejection of Claims 34-38, 43-51 and 58 and 59***

Claims 34-38, 43-51, 58 and 59 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Dunphy. This rejection is respectfully traversed.

Claim 34 is amended to more specifically define the invention. Claim 34, as amended, is directed to a method of archiving files. In relevant part, claim 34, like claim 1, recites "detecting an instruction by an operating system to perform an operation on an operating file. Dunphy does not teach, suggest or disclose such a method.

Dunphy is directed to a data storage and protection device used for data backup. As illustrated in Figure 1, Dunphy depicts an operating system 19, application programs 8 and file system 9. Data file monitor 11 is interposed between application programs 8 and file system 9. Data file monitor 11 intercepts communications between application programs 8 and file system 9. Data file monitor 11 reviews the communication to determine whether it relates to a data file that the user has selected for monitoring. If the data file is one to be monitored, it is determined whether the communication results in a change in content of a data file. If data file change is detected, data file monitor 11 extracts data file status and activity information from the received communications and uses this data to build an event log 12. Data file monitor 11 also determines whether the communication requests an operation that changes the contents of the data file, i.e., would cause a loss of data. If so, then the data file is saved. See Column 3 line 49 to column 4, line 21.

Unlike the method of claim 34, Dunphy does not detect an instruction *by an operating system*. Dunphy, much like Koshisaka and other references of record, is concerned with communications from the application programs themselves. As

Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

explained in detail in connection with the discussion of Koshisaka above, the present invention as defined by claim 34 performs file capture and file manipulation based on instructions from the operating system. This is a significant advance because it allows file capture before the file operation is performed, for example. Dunphy is limited to addressing files for which an operation that changes the file content is specified, e.g., a delete or modify operation. Dunphy would be useless against operations that do not intentionally change file content but that may corrupt file content such as file open operations or file rename operations.

Since claim 34 responds to instructions by the operating system, all operations will invoke the file archiving procedure of claim 34.

In view of the foregoing, it is apparent that claim 34 would not have been obvious in view of Dunphy. It follows that claim 34 is properly allowable. Dependent claims 36-38 and 43 are likewise properly allowable.

With respect to claim 35 it is properly allowable for the reasons set forth above for the allowability of claim 34 from which it depends. Claim 35 is further allowable because Dunphy does not teach or suggest storing an archive file prior to the operation being performed on the operating file as required by claim 35.

The Office Action relies on column 4, lines 30-38 of Dunphy as teaching that the archive file is stored prior to performing an operation the operation file. However, Dunphy simply does not stand for that proposition. The cited passage of Dunphy addresses creating an entry in an event log 12. The event log 12 includes a list of all data files that have been subject to change. Consequently, the event log does not contain the content of the data files. Furthermore, Dunphy teaches that entries in event log 12 are made *after* the changes occur. "For each of these data file changes, the data file manager 11 creates an entry in event log 12 that identifies the data directory/data file, the nature of the change, extend of the data file, the time that this change occurred and any other pertinent administrative information." Column 4, lines 32-36. Since the entry records the time that the change occurred, by definition, the entry is made subsequent to the operation that creates the data file change. Consequently, claim 35 would not have been obvious in view of Dunphy and is properly allowable.

Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

With respect to claim 44, it requires that the archive pass through two storage locations before ending up in permanent storage (its third storage location). The Office Action cites to column 4, lines 25-67 of Dunphy as teaching the method of claim 44. However, the cited portion of Dunphy does not provide such a teaching.

Lines 24-38 of Dunphy discuss creation of an event log 12. Event log 12 is not an archive file. Rather it is a collection of data that includes identifying information about a file. The closest thing that Dunphy teaches to an archive file is the data file saved in stash can 13. However, Dunphy does not teach or suggest moving that data file from stash can 13 to an intermediate storage location and subsequently to a permanent storage location as required by claim 44. Accordingly, Dunphy would not have rendered the subject matter of claim 44 obvious to the skilled artisan. It follows that claim 44 is properly allowable.

#### ***Rejection of Claims 4-8 and 13-14***

Claims 4-8 and 13-14 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Koshisaka in view of Dunphy. This rejection is respectfully traversed because neither Koshisaka nor Dunphy, taken alone or in combination, teach suggest or disclose all of the limitations of the claims.

As discussed above in connection with the rejection of claim 1, Koshisaka does not teach a method for archiving files including the step of detecting an instruction by an operating system to perform an operation on an operating file. Koshisaka detects instructions from the API. Likewise, Dunphy fails to teach that it archives files responsive to the detection of instructions by an operating system. Therefore, claims 4-8 and 13-14 are properly allowable.

With respect to claim 5, both Koshisaka and Dunphy fail to teach that the archive file includes pointers directed to one or more storage locations that contain portions of the operating file. The Office Action asserts that database 14 of Dunphy is "the second storage location" having different entries of a file. This assertion misses the mark of the complete content of claim 5. First, claim 5 states that the each of one or more storage locations contains a portion of the operating file and that the archive file includes

Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

pointers directed to those storage locations. Accordingly, claim 5 requires an archive file and it requires one or more storage locations. Dunphy does not disclose an archive file containing pointers. In fact, the Office Action does not even attempt to read any teaching of Dunphy on this limitation of claim 5. Accordingly, it is readily apparent that the subject matter of claim 5 would not have been obvious to the skilled artisan and that claim 5 is properly allowable.

### ***The Rejection of Claims 39-42***

Claims 39-42 were rejected under 35 U.S.C. § 103(a) as unpatentable over Dunphy in view of Midgley et al., U.S. Patent No. 5,608,865 (hereinafter Midgley). This rejection is respectfully traversed.

Claims 39-42 depend from claim 34. Like claim 34, each of claims 39-42 are directed to a method of archiving files including the step of detecting an instruction by an operating system to perform an operation on an operating file. As discussed above in arguments for the allowability of claim 34, Dunphy does not teach, suggest or disclose such a method.

Midgley adds nothing to the teachings of Dunphy that would have rendered the subject matter of claims 39-42 obvious. Midgley teaches systems and methods for providing continuous backup of data stored on a computer network. The systems include a synchronization replication process that replicates selected source data files on the system, thereby creating a corresponding set of replicated data files known as target data files. In addition to the synchronization replication process, the system includes a dynamic replication process that includes a plurality of agents which monitor file access operations for a server on the system. See Midgley, column 1, line 65 – column 2, line 11.

According to Midgley, the agents can monitor the activities of each of the servers to detect when a user changes one of the imaged or replicated files. The agent processes can then create a record of the changes made to a particular file and store that record within a journal file that tracks the changes made by the user. See Midgley et al., column 7, lines 46-55.

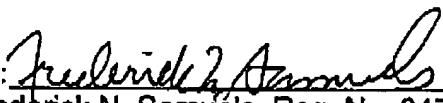


Appl. No. 09/957,459  
Amendment dated May 24, 2005  
Reply to Office Action dated February 23, 2005

There is nothing in Midgley that teaches the skilled artisan to detect an instruction by an operating system to perform an operation on an operating file and to create an archive file and store the archive file in a first storage location temporally proximate to the operation being performed on the operating file responsive to that instruction. Accordingly, neither Midgley nor Dunphy, taken alone or in combination, teach suggest or disclose a method of archiving as defined in claims 39-42. Accordingly, claims 39-42 are properly allowable.

In view of the above Remarks, Applicants submit that all of the claims of the present application are allowable and that the application is in condition for allowance. Reconsideration of the rejection and a favorable action on the merits are respectfully requested.

Respectfully submitted,  
CAHN & SAMUELS, L.L.P.

By:   
Frederick N. Samuels, Reg. No. 34715  
2000 P St., NW, Ste. 200  
Washington, D.C. 20036  
Telephone: (202) 331-8777  
Fax: (202) 331-3838

May 24, 2005